(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 887 731 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
30.12.1998 Bulletin 1998/53

(51) Int Cl.[6]: G06F 11/14

(21) Application number: 98305061.8

(22) Date of filing: 26.06.1998

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 27.06.1997 US 884097

(71) Applicant: Sun Microsystems, Inc.
Palo Alto, California 94303-4900 (US)

(72) Inventor: Molin, Hossein
San Francisco, California 94131 (US)

(74) Representative: Harris, Ian Richard et al
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Membership in an unreliable distributed computer system

(57) Multiple nodes can concurrently gain membership in a cluster of nodes of a distributed computer system by broadcasting reconfiguration messages to all nodes of the distributed computer system. In response to a reconfiguration request resulting from a node petitioning to join a cluster or a node leaving the cluster, each node determines to which nodes of the distributed computer system the node is connected, i.e., which are sending reconfiguration messages which the node receives. In addition, if multiple nodes fail substantially simultaneously, each node which continues to operate does not receive a reconfiguration message from each of the failed nodes and the failed nodes are omitted from the proposed new cluster. Thus, multiple simultaneous failures are processed in a single reconfiguration. Each of the member nodes of the proposed cluster determine the membership of the proposed cluster and broadcast a reconfiguration message to all proposed member nodes and collects similar messages. If all reconfiguration messages agree, the proposed cluster is accepted. In the case in which one or more nodes leave the cluster, quorum is established in the new cluster relative to the old cluster.

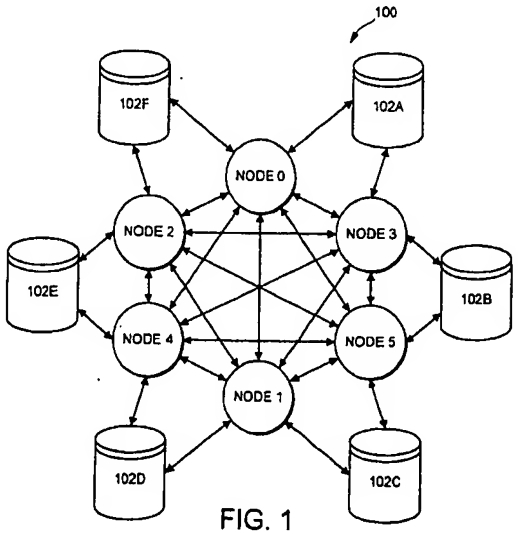

FIG. 1

EP 0 887 731 A1

Description

FIELD OF THE INVENTION

The present invention relates to distributed computer systems and, in particular, to a particularly efficient mechanism by which membership in the distributed computer system can be determined in the presence of computer system failures.

BACKGROUND OF THE INVENTION

Distributed computers systems rival and even surpass processing capabilities of supercomputers which represented the state of the art even just a few years ago. Distributed computer systems achieve such processing capacity by dividing tasks into smaller components and distributing those components to member computers of the distributed computer system, each of which processes a respective component of the task while other member computers simultaneously process other components of the task. Larger distributed computer systems promise ever increasing processing capacity at ever decreasing cost.

While distributed computer systems provide excellent processing capacity, such systems are particularly susceptible to computer hardware and software failures. Distributed computer systems have multiple computers with multiple, redundant components such as processors, memory and storage devices, and system software and further include communications media connecting the multiple member computers of the distributed computer system. Failure of any of the many constituent components of the distributed computer system can result in unavailability of the distributed computer system. Accordingly, a very important component of any distributed computer system is the ability of the system to tolerate individual or multiple, simultaneous faults. Such fault tolerance of a distributed computer system makes such a system more reliable than most single computers. Specifically, failure of a substantial portion of the distributed computer system is tolerated and processing by the distributed computer system, while diminished in capacity, continues.

In general, distributed computer systems must meet a number of criteria to properly tolerate faults and to functional adequately. First, all constituent computers of the distributed computer systems, which are sometimes referred to as "nodes," must agree regarding which of the nodes are members of a cluster. A cluster is generally a number of nodes of a distributed computer system which collective cooperate to perform distributed processing. If nodes of a distributed computer system disagree as to the membership of the cluster, nodes can also disagree as to which nodes have a quorum and therefore have access to shared resources and data. The likelihood for simultaneous, inconsistent access of the shared resources and data, and therefore corruption of the data, is great. Second, no single-point failure within a cluster can result in complete unavailability of the cluster. Such susceptibility to failure is generally unacceptable. Third, nodes of a cluster which has a quorum are never in disagreement regarding the state of the cluster. A cluster which has a quorum has exclusive access to resources which the nodes of the cluster would otherwise share with other nodes of the distributed computer system. And fourth, isolated or faulty nodes of a cluster must be removed from the cluster in a finite period of time, e.g., one minute.

Some currently available distributed computer systems can tolerate at most one failure of any node or communications link of the system at one time and can tolerate consecutive failure of every node but one. The ability to tolerate multiple, simultaneous failures in a distributed computer system greatly improves the reliability of such a distributed computer system.

SUMMARY OF THE INVENTION

Particular and preferred aspects of the invention are set out in the accompanying independent and dependent claims. Features of the dependent claims may be combined with those of the independent claims as appropriate and in combinations other than those explicitly set out in the claims.

In accordance with the present invention, multiple nodes can join a cluster simultaneously. Specifically, one or more nodes petitioning to join the cluster each determine to which nodes of the distributed computer system the nodes are connected, i.e., which are sending messages which the petitioning nodes receive, regardless of membership of each such node in the current cluster.

The petitioning nodes send a reconfigure message proposing a new cluster which includes as members all nodes to which the petitioning node is connected.

The proposed cluster can include as members nodes which are connected to the petitioning node and which are not members of the current cluster. Accordingly, more than one node can join the cluster in a single reconfiguration, thereby reducing the number of times a cluster must be reconfigured when multiple nodes are ready to join the cluster substantially simultaneously. Such is possible if multiple nodes are unavailable due to failure of a single communications link which is subsequently revived. Each node receiving the reconfigure message, referred to as a petitioned node, similarly determines all other nodes to which the node is connected and responds with reconfigure message which proposes a respective new cluster including all such nodes. The petitioning and petitioned nodes collect all reconfiguration messages and if all the reconfiguration messages unanimously propose the same proposed cluster, the proposed cluster is accepted as new. Thus, unanimous agreement as to the membership of the cluster is assured.

Further in accordance with the present invention, multiple nodes can leave a cluster simultaneously. Failure to receive messages from a particular node in a predetermined period is detected as a failure of the node. In response to the detected failure, the node detecting the failure sends a reconfigure message. Each node receiving the reconfigure message broadcasts in response thereto a reconfigure message to all nodes and determines from which nodes a reconfigure message is received. Thus, each node determines to which other nodes the node is operatively connected and configures a proposed new cluster which includes as members the connected nodes. If multiple nodes fail substantially simultaneously, each node which continues to operate does not receive any messages from each of the failed nodes and the failed nodes are omitted from the proposed new cluster. Thus, multiple simultaneous failures are processed in a single reconfiguration.

Since the failure of a node can be either a failure of the nodes itself or the communications link connecting the node to the remainder of the distributed computer system, the proposed new cluster is not accepted as the new cluster unless the proposed new cluster can establish a quorum relative to the previous member of the cluster. If the previous cluster had only two member nodes, quorum is established by a race mechanism. If the two member nodes of the previous cluster do not share a quorum device, an alternative mechanism is used to establish quorum. If the previous cluster had more than two member nodes, quorum is established by a vote mechanism in which one of the member nodes of the previous cluster is designated the crown prince to resolve quorum votes which result in a tie.

Accordingly, a distributed computer system in accordance with the present invention can tolerate simultaneous failure of up to one-half of the member nodes of a cluster. Failure of more than one-half of the member nodes of the cluster prevent the cluster from achieving a quorum. However, since quorum is established relative to the previous membership of the cluster and not relative to all nodes of the distributed computer system, the distributed computer system can tolerate a series of multiple-node failures as long as each multiple-node failure includes failure of no more than one-half of the nodes surviving the previous multiple-node failure until only one node remains operative. The distributed computer system according to the present invention is therefore particularly robust and improves significantly the likelihood that the functionality provided by the distributed computer system will continue to be provided despite multiple simultaneous, or a series of multiple simultaneous, failures.

BRIEF DESCRIPTION OF THE DRAWINGS

Exemplary embodiments of the invention are described hereinafter, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a distributed computer system in accordance with the present invention.
Figure 2 is a block diagram of two nodes of the distributed computer system of Figure 1 which share a number of devices and each of which includes a cluster membership monitor in accordance with the present invention.
Figure 3 is a block diagram of a cluster membership monitor of Figure 2.
Figure 4 is a logic flow diagram illustrating the petitioning of a node to join a cluster in the distributed computer system of Figure 1 in accordance with the present invention.
Figure 5 is a logic flow diagram illustrating the processing of nodes in response to the petitioning shown in Figure 4 to determine membership in a new cluster in accordance with the present invention.
Figure 6 is a logic flow diagram illustrating the leaving of a node from a cluster in accordance with the present invention.
Figure 7 is a logic flow diagram illustrating negotiation for quorum based on the previously current cluster membership in accordance with the present invention.
Figure 8 is a logic now diagram illustrating a race for quorum in response to one node leaving a cluster having two member nodes.
Figure 9 is a logic flow diagram illustrating a vote for quorum in response to one or more nodes leaving a cluster having more than two member nodes.
Figure 10 is a block diagram showing individual threads of the cluster membership monitor of Figure 3 according to one embodiment.

DETAILED DESCRIPTION

In accordance with the present invention, membership in a cluster of nodes in the distributed computer system is determined in a way which permits multiple nodes to simultaneously join or leave the cluster. As a result, the distributed computer system continues to provide service in spite of multiple simultaneous node failures.

Figure 1 shows an illustrative example of a distributed computer system 100 which includes nodes 0-5. Nodes 0-5 are fully interconnected, i.e., distributed computer system 100 includes a direct communications link between each of nodes 0-5 and each other of nodes 0-5. Distributed computer system 100 also includes a number of storage devices 102A-F, each of which serves as a quorum device in one embodiment. Storage device 102A is connected between and shared by nodes 0 and 3. Storage device 102B is connected between and shared by nodes 3 and 5. Storage device 102C is connected between and shared by nodes 5 and 1. Storage device 102D is connected between and shared by nodes 1 and 4. Storage device 102E is con-

nected between and shared by nodes 4 and 2. Storage device 102F is connected between and shared by nodes 2 and 0. Nodes 0-5 are described in greater detail below.

Cluster membership is determined by each of nodes 0-5 individually in such a manner that each node arrives at the same result and multiple, simultaneous failures are detected and properly handled. Each of nodes 0-5 includes a cluster membership monitor (CMM) which is a computer process executing within each of nodes 0-5. To facilitate appreciation of the present invention, a number of hardware components of nodes 0-5, and therefore the operating environment for each of the CMMs, are described.

Figure 2 shows nodes 0 and 3. Each of nodes 1-5, including node 3, are directly analogous to node 0 and the following description of node 0 is equally applicable to each of nodes 1-5. Node 0 includes one or more processors 202A, each of which retrieves computer instructions from memory 204A through an interconnect 206A and executes the retrieved computer instructions. In executing retrieved computer instructions, each of processors 202A can retrieve data from and write data to memory 204A and any and all of shared storage devices 102A and 212A-C through interconnect 206A. Interconnect 206 can be generally any interconnect mechanism for computer system components and can be, e.g., a bus, a crossbar, a mesh, a torus, or a hypercube. Memory 204A can include any type of computer memory including, without limitation, randomly accessible memory (RAM), read-only memory (ROM), and storage devices which use magnetic and/or optical storage media such as magnetic and/or optical disks. Shared storage devices 102A and 212A-C are each a storage device or an array of storage devices which can be simultaneously coupled to two or more computers. As shown in Figure 2, each of shared storage devices 102A and 212A-C is coupled both to interconnect 206A of node 0 and to interconnect 206B of node 3. Each of shared storage devices 102A and 212A-C is accessed by each of nodes 0 and 3 as a single device although each of shared storage devices 102A and 212A-C can be an array of storage devices. For example, any of shared storage devices 102A and 212A-C can be a SPARC Storage Array available from Sun Microsystems, Inc. of Mountain View, California.

Sun, Sun Microsystems, and the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Each of shared storage devices 102A and 212A-C can be reserved by either node 0 or node 3. For example, any of processors 202A can issue control signals through interconnect 206A to shared storage device 212C which cause reservation of storage device 212C. In response to the control signals, shared storage device 212C determines whether shared storage device 212C is already reserved as represented in the physical state of shared storage device 212C, e.g., in the state of a flag or an identification of the holder of the current reservation as represented in a register of shared storage device 212C. If shared storage device 212C is not currently reserved, shared storage device 212C changes its physical state to indicate that shared storage device 212C is now reserved by node 0. Conversely, if shared storage device 212C is currently reserved, shared storage device 212C sends through interconnect 206A to processors 202A signals which indicate that the attempted reservation is refused.

In addition, each of processors 202A can issue control signals to a network access device 208A which cause network access device 208A to transfer data through network 210 between network access device 208A of node 0 and network access device 208B of node 3 in a conventional manner. Network 210 includes all of the communications links between nodes 0-5 shown in Figures 1 and 2. In one embodiment, network 210 (Figure 2) is the well-known Ethernet network and network access devices 208A and 208B are conventional Ethernet controller circuitry.

Node 0 includes a cluster membership monitor (CMM) 220A which is a computer process executing in processors 202A from memory 204A. CMM 220A implements a state automaton which includes representation of the state of node 0 with respect to distributed computer system 100 (Figure 1) and of a current cluster of distributed computer system 100. CMM 220A is shown in greater detail in Figure 3 and includes a number of fields which collectively represent the state of a cluster of nodes 0-5 (Figure 1). A field is data which collectively represent a component of information. Specifically, CMM 220A (Figure 3) includes an identification field 302, a cluster size field 304, a cluster vector field 306, a next cluster size field 308, and a next cluster vector field 310.

Identification field 302 includes data which uniquely identifies node 0 and distinguishes node 0 from nodes 1-5 (Figure 1). The data stored in identification field 302 (Figure 3) are sometimes collectively referred to herein as the identifier of node 0. Cluster size field 304 (Figure 3) includes data which specify a number of nodes included in the cluster to which node 0 is a member. Cluster vector field 306 includes data which identify each member node of the cluster to which node 0 is a member. Accordingly, cluster vector field 306 includes the identifier of node 0 and can include the identifiers of each of nodes 1-4 (Figure 1). Next cluster size field 308 and next cluster vector field 310 collectively represent a state of a prospective cluster during reconfiguration as described below and are analogous to cluster size field 304 and cluster vector field 306, respectively.

When CMM 220A (Figure 2) of node 0 is initialized, CMM 220A attempts to join a cluster which includes any

of nodes 1-5 (Figure 1) according to the steps of logic flow diagram 400 (Figure 4). Processing according to logic flow diagram 400 begins in step 402 in which CMM 220A (Figure 3) initializes cluster size field 304 to zero and cluster vector field 306 to represent an empty set to indicate that no nodes are currently a member of the current cluster. Processing transfers to step 404 (Figure 4) in which CMM 220A (Figure 3) broadcasts a reconfiguration message to nodes 1-5. A reconfiguration signal generally includes a message type field which indicates that the message is a reconfiguration message and includes the identifier of the node sending the reconfiguration message and the cluster size and vector fields of the node sending the reconfiguration message. CMM 220A broadcasts the reconfiguration message to all nodes which are potentially members of a new cluster, i.e., to nodes 1-5 (Figure 1), regardless of each node's membership in any current clusters.

In step 406 (Figure 4), to which processing transfers from step 404, CMM 220A (Figure 3) waits for a predetermined period of time to receive reconfiguration messages from nodes 1-5. In one embodiment, the predetermined period of time is thirty seconds. As described in more detail below with respect to step 504 (Figure 5), each member node of a cluster responds to a reconfiguration message received from a non-member node by broadcasting a responding reconfiguration message. By waiting to receive reconfiguration messages from all nodes, CMM 220A (Figure 3) determines which, if any, of nodes 1-5 are operative and in communication with node 0. When CMM 220A has received reconfiguration messages from all of nodes 1-5 or when the predetermined period of time has expired, whichever occurs first processing transfers to step 408 (Figure 4). In step 408, CMM 220A (Figure 3) updates next cluster size field 308 and next cluster vector 310 to represent a cluster which includes node 0 and all nodes from which CMM 220A receives a reconfiguration message in step 406 (Figure 4). Thus, in steps 406 and 408, CMM 220A (Figure 3) builds a prospective cluster which includes all nodes which appear to be operative and properly connected to node 0.

It should be noted at this point that multiple nodes can join a cluster in a single reconfiguration. For example, node 2 (Figure 1) can perform the steps of logic flow diagram 400 (Figure 4) while node 0 performs the steps of logic flow diagram 400 concurrently and independently. Accordingly, reconfiguration messages broadcast by nodes 0 and 2 in independent, analogous performances of step 404 (Figure 4) are received by nodes 0 (Figure 1) and 2 in independent, analogous performances of step 406 (Figure 4). Accordingly, nodes 0 and 2 include each other in a prospective new cluster in independent, analogous performances of step 408 (Figure 4).

In test steps 410 (Figure 4) and 412, CMM 220A (Figure 3) determines whether the prospective cluster is proper. Specifically, in test step 410 (Figure 4), CMN 220A (Figure 3) compares the cluster size represented in cluster size field 304 to a value of one to determine whether any node other than node 0 is a member of the prospective cluster. If the cluster size is greater than one, processing transfers to step 414 (Figure 4) which is described below. Conversely, if the cluster size is not greater than one, processing transfers to test step 412.

In test step 412, CMM 220A (Figure 3) determines whether node 0 is isolated, i.e., whether all communications links between node 0 and other nodes of distributed computer system 100 (Figure 1) have failed. If node 0 is not isolated but is instead the sole member of a cluster, node 0 can safely participate in competitions for quorum, which are described more completely below, and other nodes can subsequently join the cluster of which node 0 is the sole member. It is generally preferred to prevent isolated nodes from operating on shared data since such presents a substantial risk that such data will become corrupted by the isolated node or other nodes which are not in communication with the isolated node. However, a node which is the sole member of a cluster is permitted to continue processing.

From the perspective of CMM 220A (Figure 2) of node 0, isolation of node 0 and exclusive membership in a single-node cluster are indistinguishable. In one embodiment, the determination regarding whether node 0 is isolated requires human intervention. A human operator generally provides data, through physical manipulation of user input devices (not shown) of node 0 using conventional techniques, which indicates whether node 0 is isolated. The data can be provided before hand and stored in a node configuration field (not shown) from which CMM 220A (Figure 3) retrieves the data. Alternatively, the operator can be prompted to provide the data by CMM 220A using conventional user-interface techniques. If node 0 is isolated, processing transfers from test step 412 (Figure 4) to step 420 in which node 0 fails to join a cluster and CMM 220A (Figure 3) aborts processing in the manner described more completely below. Conversely, if node 0 is not isolated, node 0 proceeds to form a cluster to which node 0 is the sole member and processing transfers to step 414 (Figure 4).

In step 414, CMM 220A (Figure 3) requests a reconfiguration of the current cluster of distributed computer system 100 (Figure 1) by broadcasting a reconfiguration message which includes the prospective cluster size and vector represented in next cluster size field 308 (Figure 3) and next cluster vector field 310. CMM 220A broadcasts the reconfiguration message to each of nodes 1-5 which is identified in next cluster vector 310. In the context of logic flow diagram 400 (Figure 4), each such node is referred to as a petitioned node. Processing transfers to step 416 (Figure 4) in which CMM 220A (Figure 3) waits for a predetermined period of time to receive reconfiguration messages from all petitioned nodes. In one embodiment, the predetermined period of time is thirty seconds. The manner by which a petitioned node receives a reconfiguration message from CMM

220A of node 0 and replies with another reconfiguration message is described more completely below with respect to logic flow diagram 500 (Figure 5). When CMM 220A (Figure 3) receives a reconfiguration message from each petitioned node or when the predetermined period of time expires, which ever occurs first, processing of CMM 220A transfers to test step 418 (Figure 4).

In test step 418, CMM 220A (Figure 3) determines whether reconfiguration messages have been received from all petitioned nodes. If CMM 220A fails in step 416 to receive a reconfiguration message from any of the petitioned nodes, processing transfers from test step 418 to step 420 and the reconfiguration fails. In step 420, CMM 220A (Figure 3) aborts processing and does not update cluster size field 304 and cluster vector field 306 to represent the prospective cluster. After step 420, processing according to logic flow diagram 400 (Figure 4) terminates.

Conversely, if CMM 220A (Figure 3) determines in test step 418 (Figure 4) that reconfiguration messages from all petitioned nodes are received in step 416, processing transfers to test step 422. In test step 422, CMM 220A (Figure 3) compares the received reconfiguration messages to determine whether all the received reconfiguration messages represent exactly the same cluster, i.e., whether all received reconfiguration messages agree as to cluster membership in the prospective cluster. If any of the received reconfiguration messages do not agree as to cluster membership, processing transfers from test step 422 (Figure 4) to step 420 in which the reconfiguration of the cluster fails in the manner described above. Conversely, if all received reconfiguration messages agree as to membership in the prospective cluster, processing transfers from test step 422 to step 424. In step 424, the prospective cluster is accepted and node 0 saves the prospective cluster as the current cluster by copying data from next cluster size field 308 (Figure 3) and next cluster vector field 310 to cluster size field 304 and cluster vector field 306, respectively. After step 424 (Figure 4), processing according to logic flow diagram 400 terminates.

Thus, a new cluster configuration is negotiated by broadcasting a reconfiguration message to all available node over all available communications links and receiving confirmation from each petitioned node. It is noted that broadcasting reconfiguration messages to all available nodes over all available communications links imposes a relatively heavy burden on distributed computer systems with a large number of nodes. However, reconfiguration of a cluster of the nodes should be a relatively infrequent occurrence since each node and each communications link is preferably relatively stable and reliable.

Logic flow diagram 500 (Figure 5) illustrates the processing of a petitioned node in response to receipt of a reconfiguration message when no reconfiguration is in progress. All nodes receiving the reconfiguration message perform the steps of logic flow diagram 500 generally concurrently and independently. As described above, all of nodes 0-5 are generally analogous to one another. Therefore, the following description of logic flow diagram 500 in the context of node 0 is equally applicable to performance of the steps of logic flow diagram 500 by any other petitioned ones of nodes 0-5 (Figure 1). In the context of logic flow diagram 500, node 0 is the petitioned node and the one of nodes 1-5 which sends the reconfiguration message, e.g., node 1, is referred to as the petitioning node. Processing according to logic flow diagram 500 begins in step 502 (Figure 5).

In step 502, CMM 220A (Figure 3) of the petitioned node, e.g., node 0, receives the reconfiguration message from the petitioning node. CMM 220A ascertains that the reconfiguration message is a petition to join the current cluster by determining that the petitioning node, i.e., the source of the reconfiguration message, is not a member of the current cluster. As described above, multiple nodes can petition for membership in the cluster in a single reconfiguration. Accordingly, the petitioned node can receive more than one reconfiguration message in step 502. For simplification of the following description, it is assumed that only a single node is currently petitioning for membership in the cluster.

Processing transfers to step 504 (Figure 5) in which CMM 220A (Figure 3) broadcasts a reconfiguration message to all prospective members of a prospective cluster, which includes all members of the current cluster and the petitioning node. By broadcasting the reconfiguration message, CMM 220A notifies all prospective members of the prospective cluster that node 0 is operational and connected.

In step 506 (Figure 5), to which processing transfers from step 504, CMM 220A (Figure 3) waits for a predetermined period of time to receive reconfiguration messages from all prospective members of the prospective cluster excluding the petitioning node since a reconfiguration message was previously received by the petitioned node, e.g., node 0, in step 502 (Figure 5). Specifically, reconfiguration messages received in step 506 include reconfiguration messages broadcast by other petitioned nodes in analogous, independent peformances of step 504. In one embodiment, the predetermined period of time is thirty seconds.

When reconfiguration messages have been received from all prospective members of the prospective cluster have been received by CMM 220A (Figure 3) or when the predetermined time period expires, whichever occurs first, processing transfers to step 508 (Figure 5). In step 508, CMM 220A (Figure 3) stores in next cluster size field 308 and next cluster vector field 310 data which represents a cluster whose membership includes all nodes from which reconfiguration messages are received in step 506 (Figure 5), including the petitioned node, e.g., node 0. Accordingly, next cluster size field 308 (Figure 3) and next cluster vector field 310 store data representing a prospective cluster which includes as members all nodes which are operational and which

are in communication with the petitioned node.

It is important to note that, since CMM 220A determines which of the nodes of the cluster are connected and functioning in steps 502 (Figure 5) and 506 and forms the prospective new cluster from these nodes in step 508, multiple nodes can be added to the cluster simultaneously. Thus, in steps 404-408 (Figure 4) and 502-508 (Figure 5), all member nodes of a prospective new cluster determine independently which other nodes are operative and in communication with the member nodes to thereby ascertain membership of the new, prospective cluster. Accordingly, multiple nodes can join the cluster simultaneously. It should also be noted that one or more nodes which fail to respond with reconfiguration messages which are therefore not received in independent performances of step 406 (Figure 4) or step 506 (Figure 5) by each member of the prospective cluster are excluded from membership in the prospective cluster. Accordingly, a node can join the cluster while another node leaves the cluster in a single reconfiguration of the cluster.

Steps 510-520 are generally analogous to steps 414-424 (Figure 4) in that the petitioned nodes each determine whether all other members of the prospective cluster are in unanimous agreement with respect to the membership of the prospective cluster. Specifically, processing transfers from step 508 (Figure 5) to step 510 in which CMM 220A (Figure 3) broadcasts to all members of the prospective cluster a reconfiguration message which includes data specifying the prospective cluster, i.e., specifying the number and identity of the members of the prospective cluster. In step 512 (Figure 5), CMM 220A (Figure 3) waits for a predetermined period of time to receive reconfiguration messages from all members of the prospective cluster. In one embodiment, the predetermined period of time is thirty seconds. When reconfiguration messages are received from all members of the prospective cluster or when the predetermined period of time expires, whichever occurs first, processing transfers to test step 514 (Figure 5) in which CMM 220A (Figure 3) begins to determine whether the members of the prospective cluster unanimously agree to the prospective cluster's membership.

In test step 514, CMM 220A (Figure 3) determines whether a reconfiguration message is received from every member of the prospective cluster in step 512 (Figure 5). If CMM 220A (Figure 3) fails to receive a reconfiguration message from any of the members of the prospective cluster during the predetermined time period in step 512 (Figure 5), processing transfers from test step 514 to step 516. In step 516, the petitioning node is refused membership in the cluster and the cluster remains unchanged, i.e., data stored in next cluster size field 308 (Figure 3) and next cluster vector field 310 are not moved into cluster size field 304 and cluster vector field 306. After step 516 (Figure 5), processing according to logic flow diagram 500 terminates.

Conversely, if CMM 220A (Figure 3) determines in test step 514 (Figure 5) that reconfiguration messages are received from all members of the prospective cluster, processing transfers from test step 514 to test step 518. In test step 518, CMM 220A (Figure 3) compares all received reconfiguration messages to determine whether the received reconfiguration messages specify the same cluster specified by the reconfiguration message sent by CMM 220A in step 510 (Figure 5). If any of the reconfiguration messages specifies a different cluster, agreement regarding new cluster membership is not unanimous and processing transfers to step 516 in which the petitioning node is refused membership in the cluster in the manner described above. Conversely, if all reconfiguration messages specify the same cluster, agreement regarding new cluster membership is unanimous and processing transfer from test step 518 to step 520.

In step 520, the petitioning node is granted membership in the cluster and the prospective cluster is made current by copying data stored in next cluster size field 308 (Figure 3) and next cluster vector field 310 into cluster size field 304 and cluster vector field 306, respectively. After step 520 (Figure 5), processing according to logic flow diagram 500 terminates.

Leaving a Cluster

Once a cluster is established, the nodes of the cluster cooperate to distribute processing and carry the distributed processing in a conventional manner and to thereby achieve the efficiencies and benefits associated with distributed processing. On occasion, it is necessary for one or more nodes to leave the cluster. For example, a node may determine that the node can no longer guarantee accurate processing and can voluntarily withdraw from the cluster. Alternatively, a node can fail and that failure can be detected by another node of the cluster to whom the failing node had been sending reconfiguration messages. It should be noted that failure of all communication links between two nodes is detected in the same manner and is therefore processed in the same manner as if the node itself had failed. The node detecting the failure initiates a reconfiguration of the cluster to form a new cluster which does not include any failed nodes. In either case, a node broadcasts a reconfiguration message to all nodes of the cluster.

Removal of a node from the cluster in response to such a reconfiguration message is illustrated by logic flow diagram 600 (Figure 6) in which processing begins in step 602. All nodes of the cluster perform the steps of logic now diagram 600 generally concurrently and independently. As described above, all of nodes 0-5 are generally analogous to one another. Therefore, the following description of logic flow diagram 600 in the context of node 0 is equally applicable to performance of the steps of logic flow diagram 600 by any other one of nodes 0-5 (Figure 1).

In step 602 (Figure 6), CMM 220A (Figure 3) of node

0 receives the reconfiguration message. Processing transfers to step 604 (Figure 6), in which CMM 220A (Figure 3) broadcasts a reconfiguration message to all nodes in the current cluster, i.e., all nodes identified in cluster vector field 306. CMM 220A of node 0 then waits for a predetermined amount of time to receive reconfiguration messages from all nodes in the current cluster in step 606 (Figure 6) to determine which of the nodes of the cluster are in communication with node 0 and operational. In one embodiment, the predetermined period of time is thirty seconds.

The node leaving the cluster sends no messages after the reconfiguration message received in step 602. Accordingly, CMM 220A (Figure 3) does not receive a reconfiguration message from the leaving node in step 606. However, processing is slightly different if one node detects failure of another node and sends a reconfiguration message to form a new cluster which excludes the failed node. In such circumstances, the former, failure-detecting node sends a reconfiguration message in lieu of receiving a reconfiguration message in step 602 (Figure 6) but performs steps 604-612 in the manner otherwise described herein. Accordingly, the failure-detecting node broadcasts a second reconfiguration message which is received in step 606 such that the failure-detecting node is included in the new, prospective cluster after expulsion of the failed node.

When CMM 220A (Figure 3) receives reconfiguration messages from all nodes in the current cluster or when the predetermined amount of time passes, whichever occurs first, processing transfers to step 608 (Figure 6). In step 608, CMM 220A (Figure 3) forms a prospective new cluster which includes all nodes from whom CMM 220A receives reconfiguration messages in step 606 (Figure 6). The prospective new cluster is represented in next cluster size field 308 (Figure 3) and next cluster vector field 310 of CMM 220A.

It is important to note that, since CMM 220A determines which of the nodes of the cluster are operational and in communication with node 0 in step 606 (Figure 6) and forms the prospective new cluster from these nodes, multiple nodes can be removed from the cluster simultaneously. In other words, the cluster negotiation mechanism according to the present invention tolerates multiple, simultaneous failures.

From step 608 (Figure 6), processing transfers to step 610 in which CMM 220A (Figure 3) negotiates a quorum for the prospective new cluster. Quorum must generally be negotiated because failure of one or more nodes of a cluster can be indistinguishable from a failure of communication links connecting the one or more nodes to the other nodes of the cluster. If a node leaves the cluster due to failure of the node itself, the leaving node generally ceases processing and does not access resources shared with the remainder of the cluster. However, if a node leaves the cluster due to failure of a communication link, the node can continue processing and can corrupt shared resources by failing to coordinate access with other nodes which continue to operate. It is therefore important that member nodes of the prospective new cluster establish a quorum before continuing processing and accessing resources shared with the leaving node or nodes.

Step 610 (Figure 6) is shown in greater detail as logic flow diagram 610 (Figure 7) in which processing begins in test step 702. In test step 702, CMM 220A (Figure 3) determines whether the current cluster, i.e., the cluster from which one or more nodes are leaving, has more than two member nodes by comparison of data stored in cluster size record 304 to data representing a value of two. If the current cluster has no more than two member nodes, processing transfers from test step 702 (Figure 7) to step 704 in which quorum is negotiated by a race for quorum. Conversely, if the current cluster has more than two member nodes, processing transfers from test step 702 to step 706 in which quorum is negotiated by a vote for quorum. As a result, a two-node cluster negotiates quorum by a quorum race since voting for quorum can lead to uncertain or undesirable results in a two-node cluster, and a cluster with more than two nodes negotiates quorum by a quorum vote since a race for quorum can lead to less than optimum conditions in a larger cluster. Determination of quorum according to each mechanism is described more completely below. After either step 704 or step 706, processing according to logic flow diagram 610, and therefore step 610 (Figure 6), completes.

From step 610, processing transfers to step 612 in which CMM 220A (Figure 3), if CMM 220A determines that the prospective new cluster has established a quorum, fences off those former member nodes of the cluster which have not achieved quorum to prevent further processing by such nodes. Specifically, CMM 220A reserves all devices shared with a former member node of the current cluster to prevent access to the shared devices by such a node. After step 610, nodes which left the cluster can no longer access devices shared between such nodes and the member nodes of the new cluster. It is noted, however, that maliciously configured nodes can corrupt devices not shared with any member node of the new cluster.

Quorum by Race

Determining quorum by a quorum race can be difficult if the two member nodes of the current cluster do not share a quorum device. Step 704 is shown in greater detail as logic flow diagrams 704 (Figure 8). Performing a quorum race according to logic flow diagram 704 can require human operator intervention.

Processing according to logic flow diagram 704 (Figure 8) begins in test step 802. In test step 302, CMM 220A (Figure 3) determines whether the member nodes of the current cluster as represented in cluster vector field 306 share a quorum device. Briefly, a quorum device is a shared device which can be reserved by any

of the nodes which share the device and is preselected as a device for use in a quorum race. CMM 220A includes a quorum database 312 which specifies which devices are quorum devices for respective pairs of nodes 0-5. By reference to quorum database 312, CMM 220A determines whether node 0 and the other of the node of the current cluster share a quorum device. If the nodes of the current cluster share a quorum device, the quorum race proceeds with step 304 (Figure 8). Conversely, if the nodes do not share a quorum device, processing transfers to step 812 which is described below.

In step 804, CMM 220A (Figure 3) attempts to reserve the quorum device shared with the other node of the current cluster. Such reservation succeeds only if the other node has not already reserved the quorum device in an analogous performance of step 804 (Figure 8). Processing transfers to test step 806 in which CMM 220A (Figure 3) determines whether reservation of the quorum device is successful. If the quorum device is successfully reserved in step 804 (Figure 8), processing transfers from test step 806 to step 808 in which quorum is established and the prospective new cluster is accepted as new by copying the data stored in next cluster size field 308 and next cluster vector field 310 into cluster size field 304 and cluster vector field 306, respectively. After step 808, processing according to logic flow diagram 704A, and therefore step 704 (Figure 7), terminates.

Conversely, if the quorum device is not successfully reserved in step 804 (Figure 8), processing transfers from test step 806 to step 810. In step 810, CMM 220A (Figure 3) aborts processing since quorum is not established. After step 810, processing according to logic flow diagram 704A, and therefore step 704 (Figure 7), terminates.

As described above, if CMM 220A (Figure 3) determines in test step 802 (Figure 8) that the member nodes of the current cluster do not share a quorum device, processing transfers to step 812. In step 812, a human computer operator selects a winner node from the member nodes of the current cluster. CMM 220A (Figure 3) prompts the human computer operator to select a winner node from a list of member nodes of the current cluster. The human computer operator generates signals identifying the winner node by physical manipulation of user-input devices using conventional user-interface techniques.

Processing transfers to test step 814 (Figure 8) in which CMM 220A determines whether node 0 is the winner node selected in step 812. If node 0 is selected as the winner node, processing transfers to step 808 in which quorum is established and the prospective new cluster is accepted as current in the manner described above. Otherwise, if node 0 is not selected as the winner node, processing transfers to step 810 in which CMM 220A (Figure 3) aborts processing since quorum is not established.

Thus, according to logic flow diagram 704 (Figure 8), quorum is determined by a simple race when the previously current cluster includes only two member nodes and the two member nodes share a quorum device.

Quorum by Vote

In step 706, which is shown in greater detail as logic flow diagram 706 (Figure 9), CMM 220A (Figure 3) of node 0, and analogous CMMs of the member nodes of the current cluster, establish quorum by vote. Processing begins in test step 902 (Figure 9) in which CMM 220A (Figure 3) compares the number of member nodes in the prospective new cluster as represented in next cluster size record 308 with one-half the number of member nodes in the current cluster as represented in cluster size record 304. If the number of member nodes in the prospective new cluster is less than one-half of the number of member nodes of the current cluster, processing transfers to step 904. Otherwise, processing transfers to test step 906 which is described more completely below. In step 904, the proposed new cluster has not established a quorum and processing by CMM 220A aborts so that resources shared with the leaving node or nodes are not corrupted. After step 904, processing according to logic flow diagram 706, and therefore step 706 (Figure 7), completes.

In test step 906, CMM 220A (Figure 3) compares the number of member nodes of the prospective new cluster and one-half of the number of member nodes of the current cluster and determines whether the prospective new cluster includes the crown prince. The crown prince is a selected one of the member nodes of the current cluster. In general, one of the member nodes of each cluster is designated as the crown prince to resolve quorum votes which result in a tie. In one embodiment, the member node with the highest relative priority is designated the crown prince of the cluster. For example, the relative priority of each node can be embedded in the node identifier stored, for example, in identification field 302 (Figure 3) of CMM 220A. In an illustrative embodiment, each node identifier is a unique number and the numerical value of each node identifier represents a relative priority, the highest of which in a given cluster identifies the crown prince of the cluster. CMM 220A determines whether the prospective new cluster includes the crown prince of the current cluster by comparison of the node identifiers stored in next cluster vector record 310 to the node identifier of the crown prince of the current cluster. If the number of member nodes of the prospective new cluster is equal to one-half the number of member nodes of the current cluster and the prospective new cluster does not include the crown prince of the current cluster, processing transfers to step 904 in which quorum is not established by the prospective new cluster as described above.

Conversely, if (i) the number of members of the prospective new cluster is greater than one-half of the

number of member nodes of the current cluster or (ii) (a) the number of members of the prospective new cluster is equal to one-half of the number of member nodes of the current cluster and (b) the prospective new cluster includes the crown prince of the current cluster, processing transfers from test step 906 to step 908. In step 908, the prospective new cluster has established a quorum and CMM 220A (Figure 3) accepts the prospective new cluster as the current cluster by copying the data stored in next cluster size field 308 and next cluster vector field 310 into cluster size field 304 and cluster vector field 306, respectively. After step 906, processing according to logic flow diagram 706, and therefore step 706, completes.

Thus, in step 706, the member nodes of the prospective new cluster independently and unanimously negotiate quorum by vote with one or more nodes which have left the cluster.

CMM in a Multi-Threaded Environment

In one embodiment, CMM 220A (Figure 3) is a multi-threaded computer process. CMM 220A includes a main thread 1002 (Figure 10), one or more sender threads 1004, one or more receiver threads 1006, a command reader thread 1008, a transition thread 1010, a communication timeout thread 1012, a keep alive thread 1014, and an abort thread 1016. Main thread 1002 processes state changes of CMM 220A as represented in fields 302-312 in the manner described above and coordinates processing of the other threads of CMM 220A. Sender threads 1004 and receiver threads 1006 send and receive, respectively, reconfiguration messages which indicate to CMM 220A which of the other nodes are operative and in communication with CMM 220A and reconfiguration messages to coordinate reconfiguration of the current cluster. Command reader thread 1008 acts as a remote procedure calling (RPC) server for applications which execute in any member node of the current cluster of which CMM 220A is a member. Transition thread 1010 processes reconfiguration of the current cluster in accordance with logic flow diagrams 400 (Figure 4), 500 (Figure 5), and 600 (Figure 6) and uses sender threads 1004 and receiver threads 1006 to send and receive, respectively, reconfiguration messages in the manner described above. Communication timeout thread 1012 monitors messages received by receiver threads 1006 and detects failure of or, equivalently, loss of communication with a member node of the current cluster. Communication timeout thread 1012 includes global fields 1018 which store data representing respective states of the member nodes of the current cluster. Keep alive thread 1014 generates reconfiguration messages and causes sender threads 1004 to send the reconfiguration messages to respective member nodes of the current cluster. Abort thread 1016 is created when processing by CMM 220A is aborted.

In one embodiment, several of the threads of CMM 220A are implemented in the kernel of the operating system of node 0 to improve performance and to simplify implementation. For example, keep alive thread 1014, communication timeout thread 1012, and receiver threads 1006 can use kernel timeout interrupts to periodically send and receive conventional heartbeat messages to periodically indicate that node 0 is operational and in communication with each of the nodes of the current cluster.

The above description is illustrative only and is not limiting.

Claims

1. A method implemented by a subject node computer of a distributed computer system for adding at least one new node to a cluster which includes at least one member node and which includes at least the subject node computer, the method comprising:

   (a) determining which connected other nodes of the distributed computer system are operative and in communication with the subject node computer;
   (b) creating a reconfigure message which specifies a proposed new cluster which includes the connected nodes;
   (c) transmitting the reconfigure message to each of the other nodes;
   (d) receiving a responsive reconfiguration message from each of the other nodes; and
   (e) determining that each responsive reconfiguration message specifies equivalent membership of the proposed new cluster to the membership specified by the reconfigure message.

2. The method of Claim 1 wherein step (a) comprises:

   waiting to receive messages from the at least one member node of the distributed computer system for a predetermined period of time; and determining that each of the at least one member node of the distributed computer system from which a message is received in the predetermined period of time is operative and in communication with the subject node computer.

3. A method implemented by a subject node computer of a distributed computer system for removing at least one leaving node from an old cluster which includes at least one member node and which includes at least the subject node computer and the leaving node, the method comprising:

   determining which connected other nodes of the at least one member node of the distributed computer system are operative and in commu-

nication with the subject node computer;
configuring a prospective new cluster which includes the connected other nodes; and
attempting to establish a quorum for the proposed new cluster relative to the old cluster.

4. The method of Claim 3 wherein the step of determining comprises:

waiting to receive messages from the at least one member node of the distributed computer system for a predetermined period of time; and determining that each of the at least one member node of the distributed computer system from which a message is received in the predetermined period of time is operative and in communication with the subject node computer.

5. The method of Claim 3 wherein the step of attempting to establish a quorum comprises:

determining a number of member nodes of the old cluster;
attempting to establish the quorum using a face mechanism upon a condition in which the number of member nodes of the old cluster is two; and
attempting to establish the quorum using a vote mechanism upon a condition in which the number of member nodes of the old cluster is greater than two.

6. The method of Claim 5 wherein the step of attempting to establish the quorum using a race mechanism comprises:

determining whether the leaving node and the subject node computer share a quorum device; and
attempting to establish the quorum using an alternative mechanism upon a condition in which the leaving node and subject node computer do not share a quorum device.

7. The method of Claim 6 wherein the alternative mechanism includes performance of the steps of:
prompting a human operator to select a winner node from a group consisting of the leaving node and the subject node computer using user-interface techniques.

8. A computer readable medium useful in association with a subject node computer which includes at least one processor and a memory, the computer readable medium including computer instructions which are configured to cause the subject node computer to add at least one new node to a cluster which includes at least one member node and which includes at least the subject node computer by performing the steps of:

(a) determining which connected other nodes of the distributed computer system are operative and in communication with the subject node computer regardless of membership of the connected nodes in the cluster;
(b) creating a reconfiguration message which specifies a proposed new cluster which includes the connected nodes;
(c) transmitting the reconfiguration message to each of the other nodes;
(d) receiving a responsive reconfiguration message from each of the other nodes; and
(e) determining that each responsive reconfiguration message specifies equivalent membership of the proposed new cluster to the membership specified by the reconfiguration message.

9. The computer readable medium of Claim 8 wherein step (a) comprises:

waiting to receive messages from the at least one member node of the distributed computer system for a predetermined period of time; and determining that each of the at least one member node of the distributed computer system from which a message is received in the predetermined period of time is operative and in communication with the subject node computer.

10. A computer readable medium useful in association with a subject node computer which includes at least one processor and a memory, the computer readable medium including computer instructions which are configured to cause the subject node computer to remove at least one leaving node from an old cluster which includes at least one member node and which includes at least the subject node computer and the leaving node by performing the steps of:

determining which connected other nodes of the distributed computer system are operative and in communication with the subject node computer regardless of membership of the connected other nodes in the old cluster;
configuring a prospective new cluster which includes the connected other nodes; and
attempting to establish a quorum for the proposed new cluster relative to the old cluster.

11. The computer readable medium of Claim 10 wherein the step of determining comprises:

waiting to receive messages from the at least

one member node of the distributed computer system for a predetermined period of time; and determining that each of the at least one member node of the distributed computer system from which a message is received in the predetermined period of time is operative and in communication with the subject node computer.

12. The computer readable medium of Claim 10 wherein the step of attempting to establish a quorum comprises:

determining a number of member nodes of the old cluster;
attempting to establish the quorum using a race mechanism upon a condition in which the number of member nodes of the old cluster is two; and
attempting to establish the quorum using a vote mechanism upon a condition in which the number of member nodes of the old cluster is greater than two.

13. The computer readable medium of Claim 12 wherein the step of attempting to establish the quorum using a race mechanism comprises:

determining whether the leaving node and the subject node computer share a quorum device; and
attempting to establish the quorum using an alternative mechanism upon a condition in which the leaving node and subject node computer do not share a quorum device.

14. The computer readable medium of Claim 13 wherein the alternative mechanism includes performance of the steps of :

prompting a human operator to select a winner node from a group consisting of the leaving node and the subject node computer using user-interface techniques.

15. A subject node computer system comprising:

at least one processor;
a memory operatively coupled to the at least one processor; and
a cluster membership module configured to cause the subject node computer system to add at least one new node to a cluster which includes at least one member node and which includes at least the subject node computer system by performing the steps of:

(a) determining which connected other nodes of the distributed computer system are operative and in communication with the subject node computer system regardless of membership of the connected nodes in the cluster;
(b) creating a reconfigure message which specifies a proposed new cluster which includes the connected nodes;
(c) transmitting the reconfigure message to each of the other nodes;
(d) receiving a responsive reconfiguration message from each of the other nodes; and
(e) determining that each responsive reconfiguration message specifies equivalent membership of the proposed new cluster to the membership specified by the reconfigure message.

16. The subject node computer system of Claim 15 wherein step (a) comprises:

waiting to receive reconfiguration messages for a predetermined period of time; and
determining that each node from which a reconfiguration message is received in the predetermined period of time is operative and in communication with the subject node computer system.

17. A subject node computer system comprising:

at least one processor;
a memory operatively coupled to the at least one processor; and
a cluster membership module configured to cause the subject node computer system to remove at least one leaving node from an old cluster which includes at least one member node and which includes at least the subject node computer system and the leaving node by performing the steps of:

determining which connected other nodes of the distributed computer system are operative and in communication with the subject node computer regardless of membership of the connected other nodes in the old cluster;
configuring a prospective new cluster which includes the connected other nodes; and
attempting to establish a quorum for the proposed new cluster relative to the old cluster.

18. The subject node computer system of Claim 17 wherein the step of determining comprises:

waiting to receive reconfiguration messages for

a predetermined period of time; and
determining that each node from which a reconfiguration message is received in the predetermined period of time is operative and in communication with the subject node computer system.

19. The subject node computer system of Claim 17 wherein the step of attempting to establish a quorum comprises:

determining a number of member nodes of the old cluster;
attempting to establish the quorum using a race mechanism upon a condition in which the number of member nodes of the old cluster is two; and
attempting to establish the quorum using a vote mechanism upon a condition in which the number of member nodes of the old cluster is greater than two.

20. The subject node computer system of Claim 19 wherein the step of attempting to establish the quorum using a race mechanism comprises:

determining whether the leaving node and the subject node computer system share a quorum device; and
attempting to establish the quorum using an alternative mechanism upon a condition in which the leaving node and subject node computer system do not share a quorum device.

21. The subject node computer system of Claim 20 wherein the alternative mechanism includes performance of the steps of:

prompting a human operator to select a winner node from a group consisting of the leaving node and the subject node computer system using user-interface techniques.

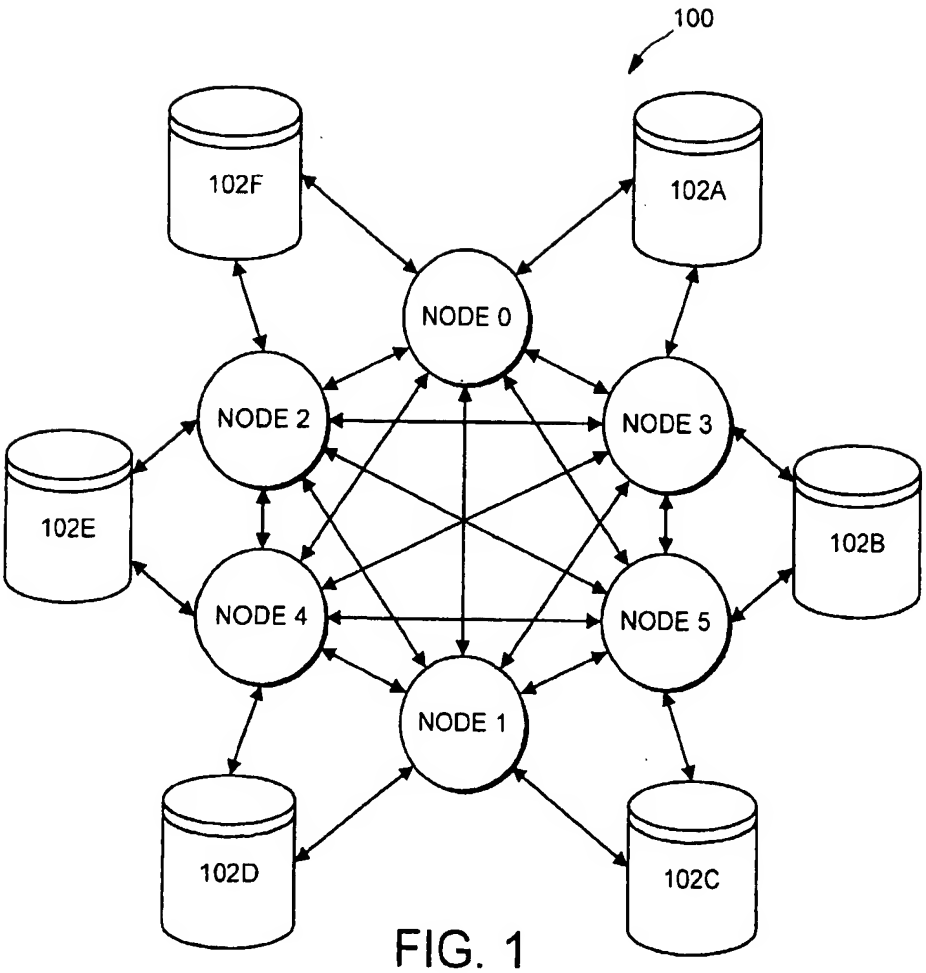
100
102F
102A
NODE 0
NODE 2
NODE 3
102E
102B
NODE 4
NODE 5
NODE 1
102D
102C

FIG. 1

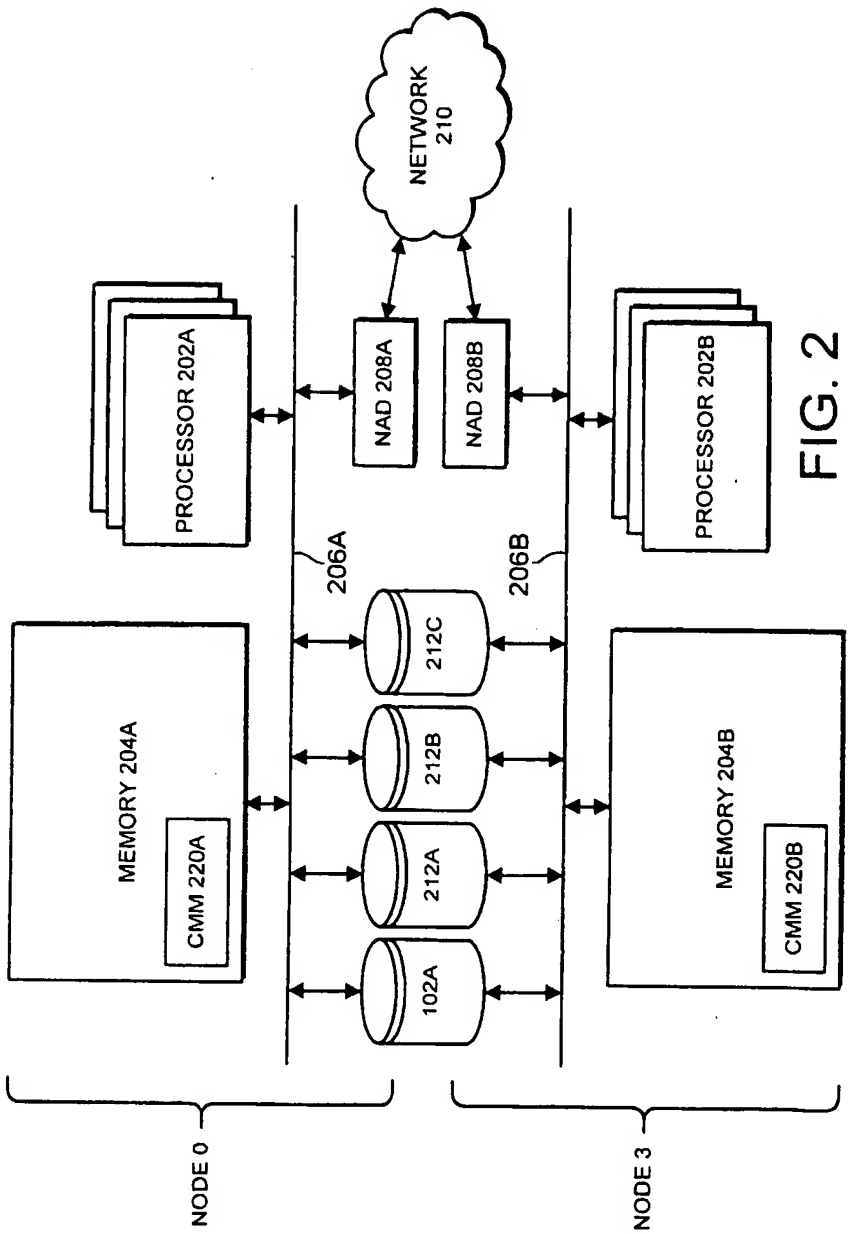
NETWORK
210
NAD 208A
NAD 208B
PROCESSOR 202A
PROCESSOR 202B
206A
206B
212C
212B
212A
102A
MEMORY 204A
CMM 220A
MEMORY 204B
CMM 220B
NODE 0
NODE 3

FIG. 2

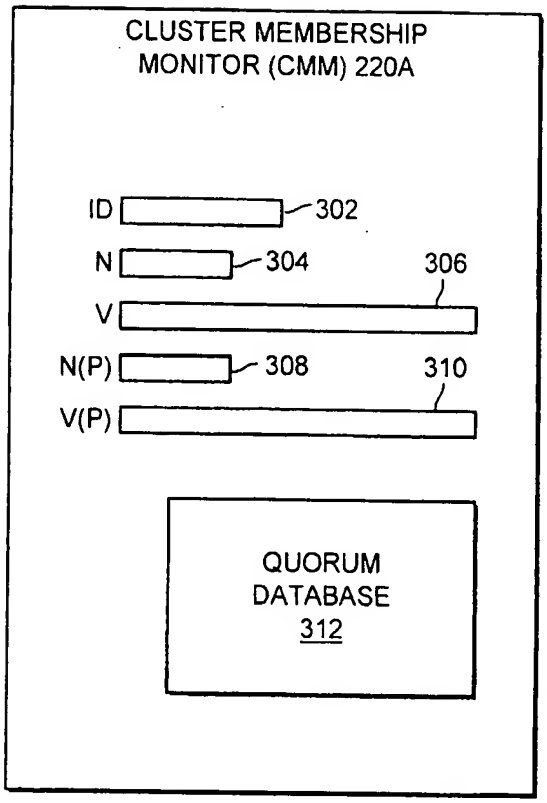
CLUSTER MEMBERSHIP
MONITOR (CMM) 220A
ID
302
N
304
306
V
N(P)
308
310
V(P)
QUORUM
DATABASE
312

FIG. 3

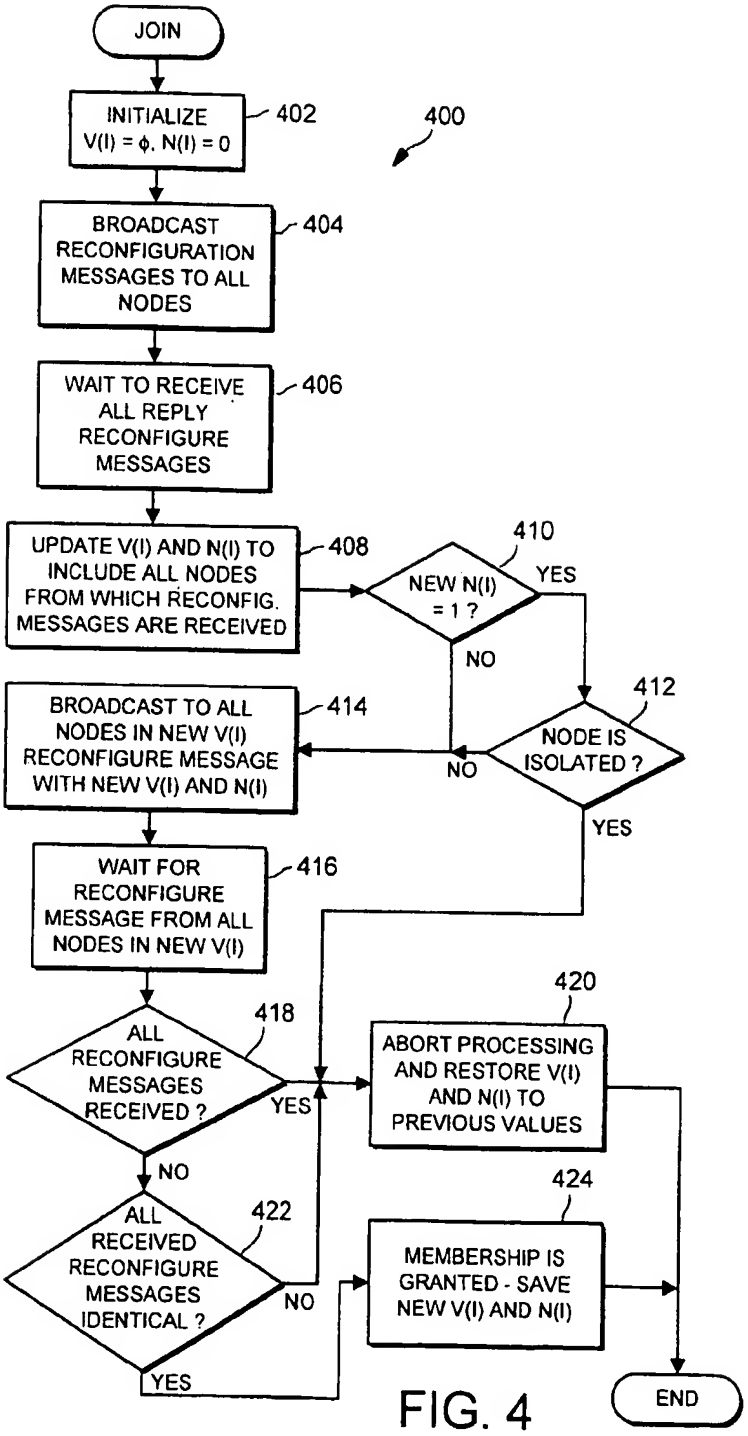
JOIN
INITIALIZE V(I) = ϕ. N(I) = 0
402
400
BROADCAST RECONFIGURATION MESSAGES TO ALL NODES
404
WAIT TO RECEIVE ALL REPLY RECONFIGURE MESSAGES
406
UPDATE V(I) AND N(I) TO INCLUDE ALL NODES FROM WHICH RECONFIG. MESSAGES ARE RECEIVED
408
NEW N(I) = 1 ?
410
YES
NO
NODE IS ISOLATED ?
412
NO
BROADCAST TO ALL NODES IN NEW V(I) RECONFIGURE MESSAGE WITH NEW V(I) AND N(I)
414
YES
WAIT FOR RECONFIGURE MESSAGE FROM ALL NODES IN NEW V(I)
416
ALL RECONFIGURE MESSAGES RECEIVED ?
418
YES
NO
ABORT PROCESSING AND RESTORE V(I) AND N(I) TO PREVIOUS VALUES
420
ALL RECEIVED RECONFIGURE MESSAGES IDENTICAL ?
422
NO
YES
MEMBERSHIP IS GRANTED - SAVE NEW V(I) AND N(I)
424
END

FIG. 4

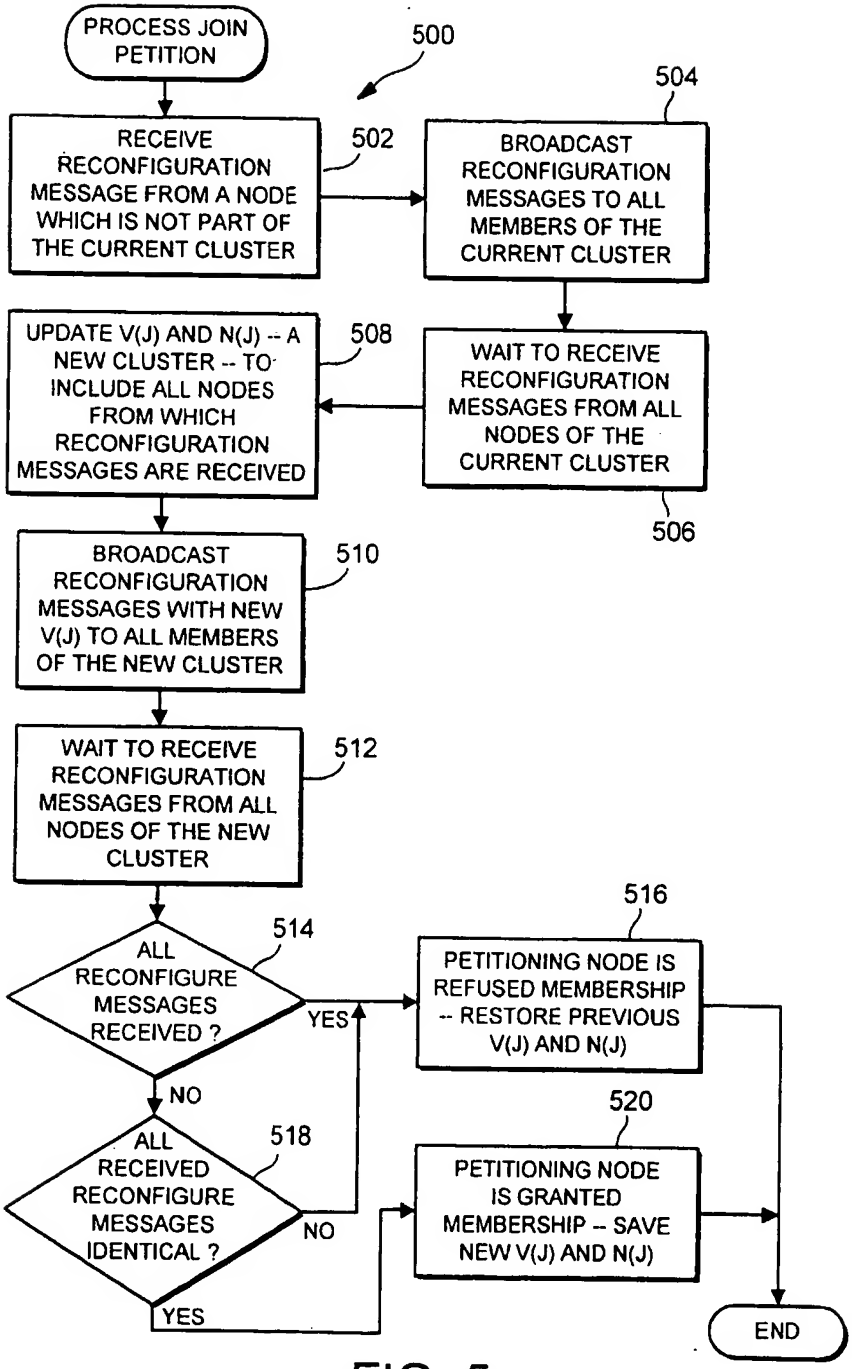
PROCESS JOIN PETITION
500
RECEIVE RECONFIGURATION MESSAGE FROM A NODE WHICH IS NOT PART OF THE CURRENT CLUSTER
502
BROADCAST RECONFIGURATION MESSAGES TO ALL MEMBERS OF THE CURRENT CLUSTER
504
WAIT TO RECEIVE RECONFIGURATION MESSAGES FROM ALL NODES OF THE CURRENT CLUSTER
506
UPDATE V(J) AND N(J) -- A NEW CLUSTER -- TO INCLUDE ALL NODES FROM WHICH RECONFIGURATION MESSAGES ARE RECEIVED
508
BROADCAST RECONFIGURATION MESSAGES WITH NEW V(J) TO ALL MEMBERS OF THE NEW CLUSTER
510
WAIT TO RECEIVE RECONFIGURATION MESSAGES FROM ALL NODES OF THE NEW CLUSTER
512
ALL RECONFIGURE MESSAGES RECEIVED ?
514
YES
NO
PETITIONING NODE IS REFUSED MEMBERSHIP -- RESTORE PREVIOUS V(J) AND N(J)
516
ALL RECEIVED RECONFIGURE MESSAGES IDENTICAL ?
518
NO
YES
PETITIONING NODE IS GRANTED MEMBERSHIP -- SAVE NEW V(J) AND N(J)
520
END

FIG. 5

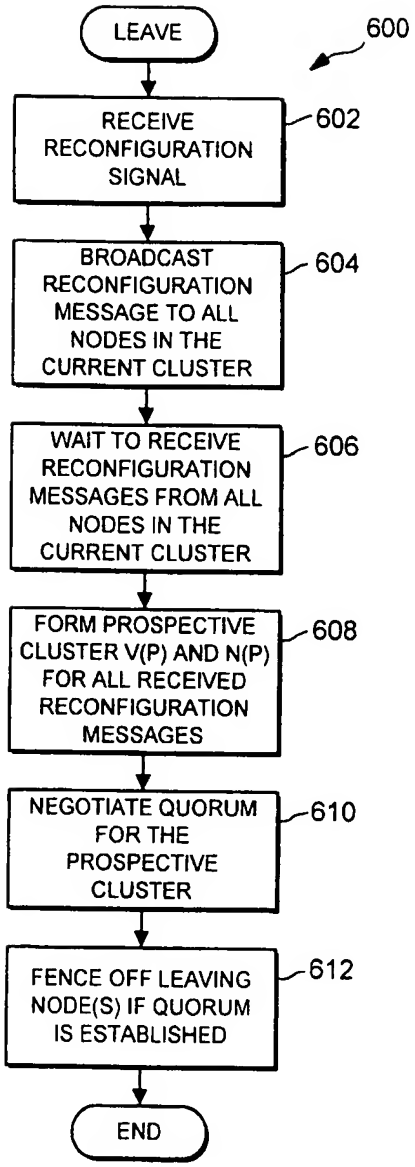
LEAVE
600
RECEIVE RECONFIGURATION SIGNAL
602
BROADCAST RECONFIGURATION MESSAGE TO ALL NODES IN THE CURRENT CLUSTER
604
WAIT TO RECEIVE RECONFIGURATION MESSAGES FROM ALL NODES IN THE CURRENT CLUSTER
606
FORM PROSPECTIVE CLUSTER V(P) AND N(P) FOR ALL RECEIVED RECONFIGURATION MESSAGES
608
NEGOTIATE QUORUM FOR THE PROSPECTIVE CLUSTER
610
FENCE OFF LEAVING NODE(S) IF QUORUM IS ESTABLISHED
612
END

FIG. 6

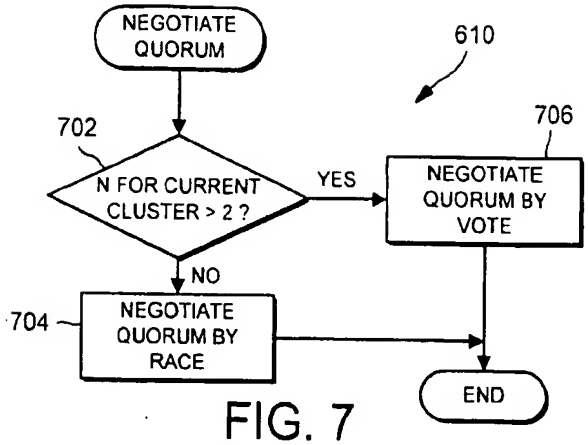
NEGOTIATE QUORUM
610
702
N FOR CURRENT CLUSTER > 2 ?
YES
706
NEGOTIATE QUORUM BY VOTE
NO
704
NEGOTIATE QUORUM BY RACE
END

FIG. 7

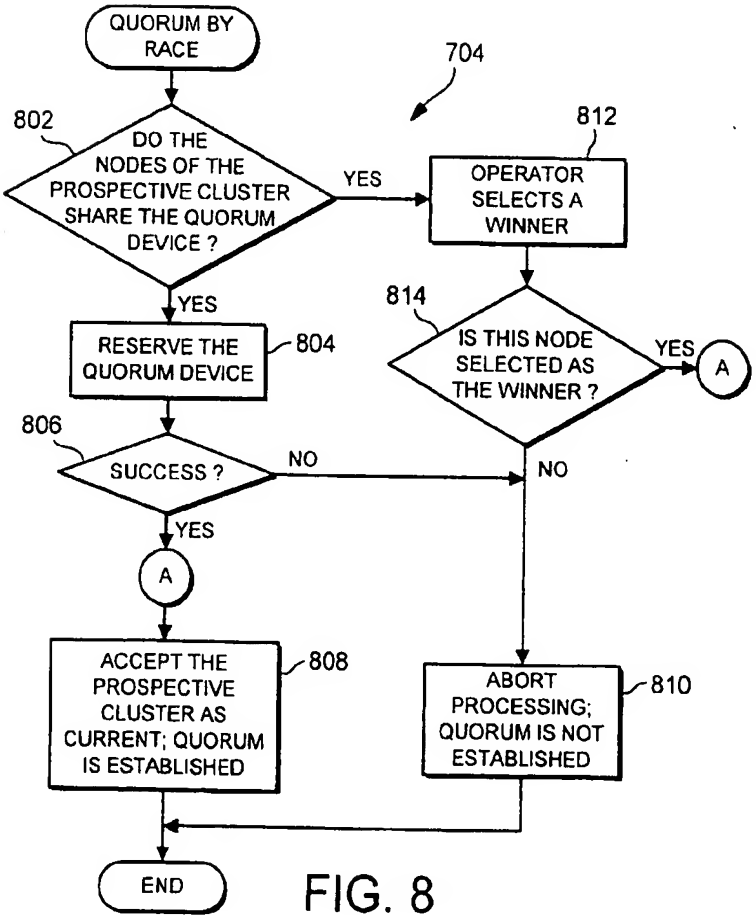
QUORUM BY RACE
704
802
DO THE NODES OF THE PROSPECTIVE CLUSTER SHARE THE QUORUM DEVICE ?
YES
812
OPERATOR SELECTS A WINNER
YES
RESERVE THE QUORUM DEVICE
804
814
IS THIS NODE SELECTED AS THE WINNER ?
YES
A
806
SUCCESS ?
NO
NO
YES
A
ACCEPT THE PROSPECTIVE CLUSTER AS CURRENT; QUORUM IS ESTABLISHED
808
ABORT PROCESSING; QUORUM IS NOT ESTABLISHED
810
END

FIG. 8

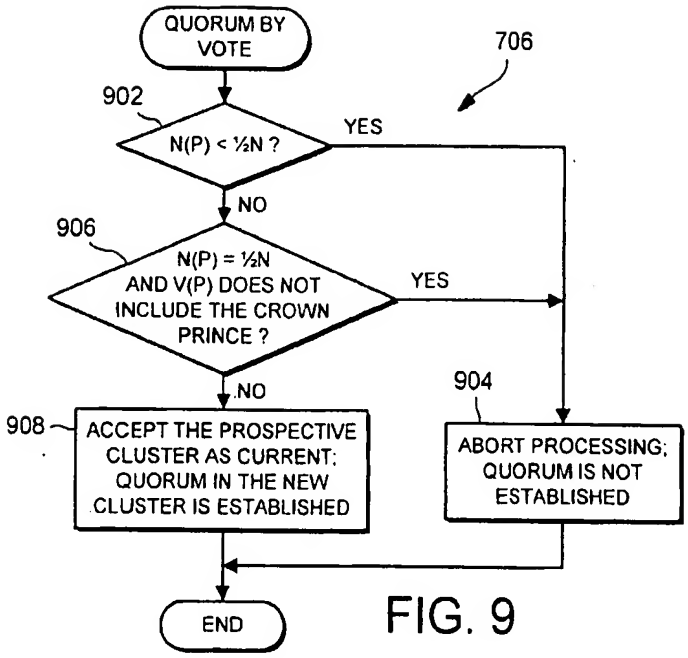
QUORUM BY VOTE
706
902
N(P) < ½N ?
YES
NO
906
N(P) = ½N AND V(P) DOES NOT INCLUDE THE CROWN PRINCE ?
YES
NO
904
908
ACCEPT THE PROSPECTIVE CLUSTER AS CURRENT; QUORUM IN THE NEW CLUSTER IS ESTABLISHED
ABORT PROCESSING; QUORUM IS NOT ESTABLISHED
END

FIG. 9

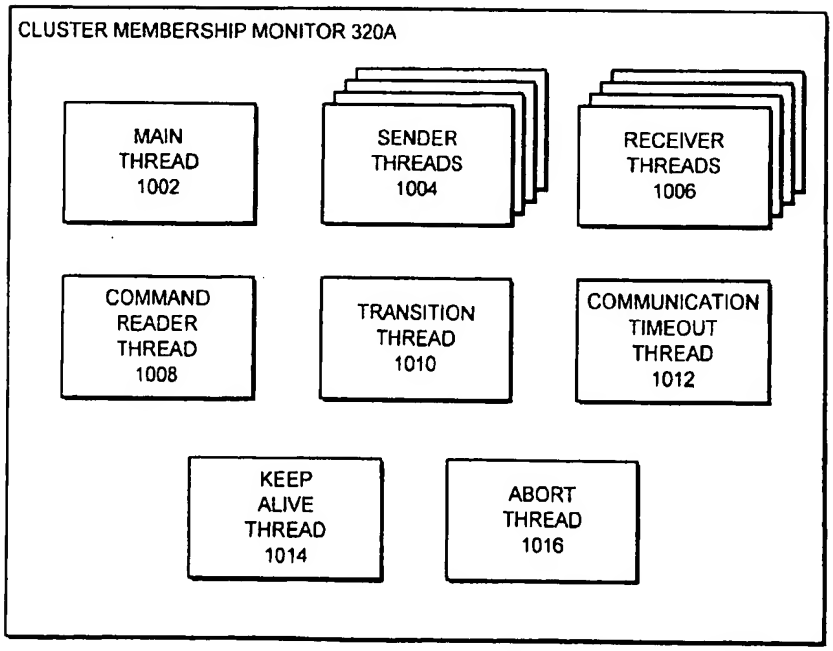
CLUSTER MEMBERSHIP MONITOR 320A
MAIN THREAD 1002
SENDER THREADS 1004
RECEIVER THREADS 1006
COMMAND READER THREAD 1008
TRANSITION THREAD 1010
COMMUNICATION TIMEOUT THREAD 1012
KEEP ALIVE THREAD 1014
ABORT THREAD 1016

FIG. 10

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 30 5061

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	EP 0 772 126 A (SUN MICROSYSTEMS, INC.) 7 May 1997 * column 3, line 52 - column 4, line 19 *	1-21	G06F11/14
A	EP 0 481 231 A (INTERNATIONAL BUSINESS MACHINES) 22 April 1992 * abstract *	2,9,16	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	13 October 1998	Cormans, G

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)